

ПЯВУ. Лекция 3.

Основы программирования.

А.М. Задорожный

Контрольные вопросы

1. Какой составной оператор (оператор, включающий другие операторы) мы рассмотрели на прошлой лекции? Что он позволяет делать?
2. Какие типы данных встречались в лекциях?
3. Что такое литерал в тексте программы? Привести примеры.

Содержание

1. Понятие цикла
2. Оператор `while`
3. Представление целых чисел в компьютере
4. Побитовые операции над целыми
5. Действительные числа и их представление в компьютере (числа с плавающей точкой)

Циклы

- Цикл (циклические вычисления) – многократ-ное выполнение одного и того же набора команд (инструкций).
- Циклы - ключевой инструмент программирования.
- Сводить решения задач к многократному выполнению одинаковых действий – важное искусство, которым должен овладеть каждый программист.

Оператор while

```
while (<булевское выражение>) // Условие продолжения  
цикла  
{  
    <инструкции> // Тело цикла  
}
```

Порядок выполнения цикла:

- Вычисляется условие.
 - Если условие истинно, выполняется тело цикла
 - Если условие ложно, оператор цикла заканчивается. Выполнение передается следующему за ним оператору.

Пример. Сумма ряда

Задача: найти сумму первых N натуральных чисел.
N – входные данные.

```
int i = 1, sum = 0; // sum – аккумулятор, i - счетчик

while(i <= N)
{
    sum = sum + i; // прибавляем к sum очередное число
    i = i + 1; // переходим к следующему числу
}

// В переменной sum содержится искомая сумма.
Console.WriteLine(sum);
```

Пример.

Проверка простоты числа

Задача: Выяснить, является ли натуральное число N простым.
 N – входные данные

```
int i = 2;           // i – счетчик  
bool prim = true;    // сначала считаем простым
```

```
while(i <= Math.Sqrt(N) && prim)  
{  
    if(N % i == 0) prim = false;  
    i = i + 1;  
}
```

// Выходные данные – переменная prim.

// Если prim истинно, то N простое. Иначе – составное.

Контрольные вопросы

1. Что такое Цикл в программировании?
2. Какой оператор C# позволяет организовать цикл?
3. Обязательно ли присутствуют круглые скобки в записи оператора цикла?
4. Что указывается в круглых скобках оператора `while`?

Представление целых чисел в компьютере

- Целое – 4 байта (32 бита)

Сложение в двоичной системе. Для 8 бит: пусть $x = 11111111$.

```
11111111
00000001
-----
00000000
```

Тогда $x + 1 = 0 \Rightarrow x = -1!$

В компьютере обычно старший бит числа определяет знак:

Если он 1, то число отрицательное,
если 0 – положительное.

Остальные биты определяют модуль числа.

Представление целых чисел в компьютере

Задачи.

1. Сколько различных значений может принимать величина типа `int`? (`int` занимает 4 байта)
2. Сколько различных отрицательных значений может принимать величина типа `int`? Какое значение минимально?
3. Сколько положительных значений может принимать величина типа `int`? Каково максимальное значение?

Целые числа и побитовые операции

- Бит – логическое значение (0 или 1)
- В C# существуют побитовые операции $|$, $&$, \sim и \wedge .
- Побитовые операции применяются независимо к каждому биту числа.

В отличие от логических, они обозначаются одинарными значками.

Целые числа и побитовые операции

| – побитовое 'или'

0001
0010
0011

& – побитовое 'и'

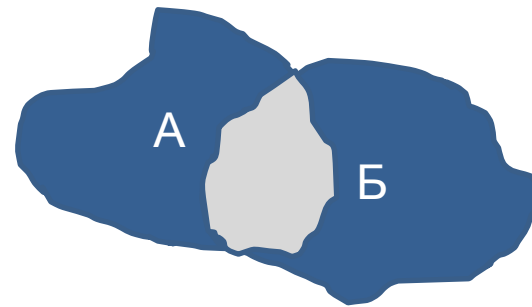
1001
1010
1000

^ – побитовое 'исключающее или' (XOR)

1001
1010
0011

~ – побитовое отрицание

~1001
0110



Побитовое представление числа

Задача. Вывести двоичное представление числа на консоль.

```
int m = 128, k = ...;           // k не больше 255
```

```
if(k & m != 0) Console.Write(1);  
else Console.Write(0);
```

```
m = m/2;
```

```
if(k & m != 0) Console.Write(1);  
else Console.Write(0);
```

```
m = m/2;
```

```
...
```

И так 8 раз.

Двоичное представление с ЦИКЛОМ

Задача. Вывести двоичное представление числа на консоль.

```
int m = 1024 * 1024 * 1024; // 2^30
if (x < 0) //Обработать знаковый бит
    Console.Write(1);
else
    Console.Write(0);

while (m != 0)
{
    if ((x & m) != 0)
        Console.Write(1);
    else
        Console.Write(0);
    m = m / 2; //Смещаем 1 вправо на 1 разряд
}
```

Упражнения

Входные данные – целочисленная переменная x .

```
int m=1;      // раньше было 2^30
while(m != 0)
{
    if((x & m) != 0)
        Console.Write(1);
    else
        Console.Write(0);
    m = 2*m;   //Смещаем 1 на 1 разряд влево. Раньше смещали вправо - m = m / 2
}
```

1. Почему цикл закончится?
2. Сколько символов будет выведено на консоль?
3. Что изменится, если условие цикла $m \neq 0$ заменить на условие $m > 0$? Почему?
4. Выполните программу, задав начальное значение $x = \text{int.MinValue}$. Обратите внимание на полученный результат.

Действительные числа

- Целых чисел на практике мало. Нужны действительные числа.
- В компьютере действительным числам соответствуют **числа с плавающей точкой** (запятой).

```
double x = 0.5; // int x = 1; bool f = true;
```


Свойства чисел с плавающей точкой

- Диапазон от 10^{200} до 10^{-200}
- $+$, $-$, $*$, $/$ - операция деления выполняется обычным образом, т.е. в результате получается дробное число.
- Библиотека математических функций:
`Math.Abs(x)`, `Math.Cos(x)`/`Math.Sin(x)`,
`Math.Exp(x)`, `Math.Log10(x)`, `Math.Pow(x, y)`,
`Math.Sqrt(x)`, ...

Особенности double

- Double – не действительные числа. (В компьютере только целые числа - коды)
- Double занимают 8 байт. Их диапазон огромен, но их число ограничено!
- Вычисления выполняются приближенно!

double x = 1.1, y = 1, z = 0.1;

$(x - (y + z)) \neq ((x - y) - z)$

- Различие составляет $\sim 8.33 \cdot 10^{-17}$

Представление с плавающей точкой

$$X = m * 10^{\text{exp}}$$

- m – мантисса
- exp – экспонента
- А так же знаковый бит.
- Диапазон определяется экспонентой;
- Точность определяется мантиссой;

Умножение и сложение с плавающей точкой

Умножение

- $M = m_1 * m_2$ здесь количество знаков увеличится
- $Exp = exp_1 + exp_2$
- **Нормализация** здесь мантисса переводится в нужный диапазон и обрезается, а экспонента компенсирует сдвиг мантиссы

Сложение

- **Выравнивание порядков**
- $M = m_1 + m_2$ здесь количество знаков увеличится
- **Нормализация** здесь мантисса переводится в нужный диапазон и обрезается, а экспонента компенсирует сдвиг мантиссы

Смешанные выражения и =

Смешанные выражения - выражения, в которых присутствуют различные числовые типы.

```
int n = 4;
```

```
double x = 3;
```

```
double y = n * x;
```

```
int m = n * x; // Синтаксическая ошибка
```

“Ловушки” в выражениях

- Порядок вычислений

$3/4*2.0$ // == 0

$2.0*3/4$ // == 1.5

- Вид операции /

$2.0/4*3$ // это не $2/12$, а $6/4$

$2.0/(4*3)$

Скобки и оптимизация

- Как упростить выражение для компьютера?

$3*x*x + 4*x + 2$. // Умножений – 3, сложений – 2.

$(3*x + 4)*x + 2$. // Умножений - 2, сложений - 2.

$3*x*x*x + 2*x*x + 3*x + 4 = > ((3*x+2)*x+3)*x + 3$

- Сохранение промежуточных результатов

$(3x^3+2x^2+3x + 4)*(3x^3+2x^2+3x + 4)$

$y = ((3*x+2)*x+3)*x + 3$.

$y*y$.

Пример.

Площадь треугольника

```
double a = 2, b = 3, c = 4.
```

Формула Герона $S = (p(p-a)(p-b)(p-c))^{1/2}$.

```
double p = (a + b + c) / 2;
```

```
double S = Math.Sqrt(p*(p-a)*(p-b)*(p-c));
```


Пример.

cos угла между двумя векторами

$$x = (x_1, x_2, x_3), \quad y = (y_1, y_2, y_3)$$

$$\cos(xy) = (x * y) / (|x| * |y|)$$

```
double xy = x1*y1+x2*y2+x3*y3;
```

```
double mx = Math.Sqrt(x1*x1+x2*x2+x3*x3),  
my = Math.Sqrt(y1*y1+y2*y2+y3*y3);
```

```
double cosXY = xy/(mx*my);
```

Число π

- Вычислить π как сумму ряда $\pi = 4/1 - 4/3 + 4/5 - 4/7 + \dots$ с точностью $\text{eps} = 0.00001$.

```
double pi = 0, eps = 0.00001;
double i = 1, alpha = 4/i - 4/(i+2);
while (alpha > eps)
{
    pi = pi + alpha;
    i = i + 4;
    alpha = 4 / i - 4 / (i + 2);
}
```

- Почему цикл закончится?
- Где аккумулятор?

Контрольные вопросы

1. Каким типом в C# представляются действительные числа?
2. Почему этот тип не представляет действительных чисел в математическом смысле?
3. Как записываются литералы, представляющие тип `double`?
4. Почему ограничен диапазон `double`? Почему ограничена точность?
5. Могут ли в одном выражении участвовать числовые данные разных типов?
6. Можно ли переменной типа `double` присвоить значение типа `int`? А наоборот?